Egon Börger (Pisa)

Reachability by Shortest Paths in a Graph

Illustrating Stepwise ASM Refinements

Dipartimento di Informatica, Università di Pisa
boerger@di.unipi.it

See Modeling Companion Ch. 4.3 (refinement variations of the PROXY programming pattern) and AsmBook Ch.3.2[1]

[1] Figures © 2003 Springer Berlin-Heidelberg, reused with permission.

The problem

Given a directed graph:

- a (finite) set $Node$
- a set $Edge$
- a distinguished node $source$

Design algorithms to compute:

- the set of nodes n which are reachable from $source$
- for each node n the 'shortest' path from $source$ to n (wrt a path measure given by the weight of edges)

Verify the algorithms proving their correctness and other properties of interest.

A sequence of refinement steps

- Computing Reachability Set: $\textsc{ShortestPath}_0$ (ground model)
- Wave Frontier Propagation: $\textsc{ShortestPath}_1$
- Nodewise Frontier Propagation to Neighborhood: $\textsc{ShortestPath}_2$
- Nodewise and Edgewise Frontier Propagation to Neighbors: $\textsc{ShortestPath}_3$
- Queue and Stack Implementation of Frontier and Neighborhoods: $\textsc{ShortestPath}_4$
- Introducing abstract weights for measuring paths and computing shortest paths: $\textsc{ShortestPath}_5$ (Moore's algorithm)
- Performing a rule optimization
- Instantiating data structures for measures and weights: a C^{++} program

Computing Graph Reachability Set

The problem:

- visit once every node which is reachable from $source$
- do not revisit nodes that have already been $visited$, so that the procedure terminates for finite graphs

Solution idea:

- starting at $source$, move along edges to reach neighbor nodes and label every reached node as $visited$
- proceed in waves, pushing in each step the visited nodes one edge further *without revisiting nodes* which have already been labeled as $visited$

From English to a mathematical model: $\textsc{ShortestPath}_0$

Initial state: only $source$ is labeled as $visited$

$\textsc{ShortestPath}_0 =$

forall $u \in visited$ **forall** $v \in neighb(u)$ -- wave propagation

if $v \notin visited$ **then** $visited(v) := true$

where

$neighb(u) = \{v \mid (u, v) \in E\}$

- Correctness Property: Each node which is reachable from $source$ is exactly once labeled as visited.
- Proof of existence claim: induction on the length of the paths starting at source.
  - induction basis: by initialization assumption
  - induction step: by applying $\textsc{ShortestPath}_0$ rule once more
- Proof of uniqueness property: rule guard ensures that no node which has already been $visited$ is relabeled.

Proving termination for $\textsc{ShortestPath}_0$

Termination Property: $\textsc{ShortestPath}_0$ terminates for finite graphs: started in the initial state, it reaches a state in which there is no longer any edge $(u, v) \in E$ where u is labeled as $visited$ but v is not.

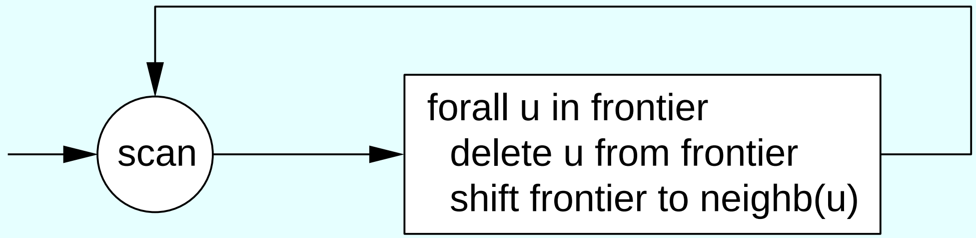
- Proof: by each step of $\textsc{ShortestPath}_0$ the (assumed to be finite) set of reachable nodes which have not yet been $visited$ decreases.

NB. The refinement steps should preserve Correctness and Termination.

First refinement: identify *frontier* of wave propagation

$frontier$ = set of nodes 'last labeled' as $visited$ (here: in the last step)

Initially $frontier = \{source\}$



$\textsc{ShortestPath}_1 =$ **forall** $u \in frontier$ -- restrict to 'last labeled'

$\quad \textsc{ShiftFrontierToNeighb}(u)$

$\quad \textsc{Delete}(u, frontier)$ -- not any more 'last labeled'

where

$\quad \textsc{ShiftFrontierToNeighb}(u) =$

$\quad\quad$ **forall** $v \in neighb(u)$ **do** $\textsc{ShiftFrontierTo}(v)$

$\quad \textsc{ShiftFrontierTo}(v) =$ **if** $v \notin visited$ **then**

$\quad\quad visited(v) := true \quad \textsc{Insert}(v, frontier)$ -- v gets 'last labeled'

Proving refinement correctness

Claim: $\textsc{ShortestPath}_1$ is a correct refinement of $\textsc{ShortestPath}_0$.

Proof. Show by induction on runs that the labeling steps of $\textsc{ShortestPath}_0$ and of $\textsc{ShortestPath}_1$

- which label some neighbor of some u as $visited$

are in 1-1 correspondence and perform the same labelings.

Induction basis $t = 1$: both machines perform one labeling step (if any) and label exactly the nodes in $neighb(source)$ as $visited$

- since by initialization $frontier = \{source\}$

Refinement correctness proof (Cont'd)

Induction step $t \Rightarrow t+1$:

- Consider any $u \in frontier$.
  - **if** $\textsc{ShortestPath}_1$ can make a labeling step with $\textsc{ShiftFrontierToNeighb}(u)$
  - **then** $\textsc{ShortestPath}_0$ can make a labeling step for $neighb(u)$

  so that the same nodes are labeled as newly $visited$.
- In step $t+1$, $\textsc{ShortestPath}_0$ applies labeling only to $u \in frontier$.
  - Proof. For every $u \notin frontier$: if u has been $visited$ by $\textsc{ShortestPath}_0$ in a step before step t, then all its $neighb$ors have been $visited$ in the next step of $\textsc{ShortestPath}_0$. Therefore $\textsc{ShortestPath}_0$ does not revisit them in step $t+1$.

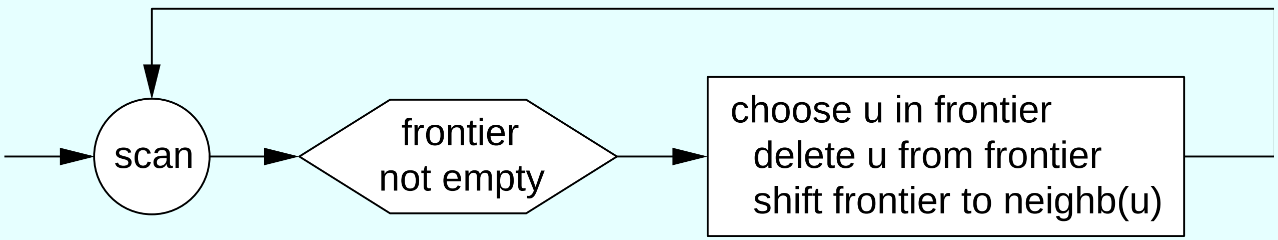
A non-empty step of $\textsc{ShortestPath}_1$ which is not a labeling step may be (only) the last one: it empties $frontier$.

Therefore *Correctness and Termination* properties are *preserved*.

Second refinement: implementing 'forall'

Idea: nodewise frontier propagation, implementing **forall** by **choose**

- Non-deterministic scheduling (to keep design space open)
- Later refinements specify constraints on $selection$ fct to guarantee properties of interest (e.g. fairness to yield completeness of node visits)



$\textsc{ShortestPath}_2 =$

if $frontier \neq \emptyset$ **then** **choose** $u \in frontier$ -- replacing **forall**

$\textsc{ShiftFrontierToNeighb}(u) \quad \textsc{Delete}(u, frontier)$

last labeled in $frontier$ is refined to mean any $visited$ node u to which $\textsc{ShiftFrontierToNeighb}(u)$ has not yet been applied

Refinement correctness for nodewise frontier propagation

Simulation Lemma. $\textsc{ShortestPath}_2$ runs with *breadth-first nodewise frontier propagation* simulate $\textsc{ShortestPath}_1$ runs.

- In other words, $\textsc{ShortestPath}_2$ with breadth-first nodewise frontier propagation is a correct refinement of $\textsc{ShortestPath}_1$.

Proof: One $\textsc{ShortestPath}_1$ step, applied to a $frontier$, corresponds to the segment of $\textsc{ShortestPath}_2$ steps which choose successively all and only the nodes from this $frontier$

- applying the same $\textsc{ShiftFrontierToNeighb}(u)$.

This is called a $(1, m)$-refinement with various m, depending on the size m of the neighborhoods which dynamically determine the frontier.

Relating frontier propagation in $\textsc{ShortestPath}_{1/2}$ runs

Slow Down Lemma. For maximal $\textsc{ShortestPath}_i$ runs $(i = 1, 2)$, i.e. where each applicable rule is eventually applied, the following holds:

1. Claim 1. For each step t and each $u \in \mathit{frontier}_t(\textsc{ShortestPath}_2)$ there exists a $t' \leq t$ such that $u \in \mathit{frontier}_{t'}(\textsc{ShortestPath}_1)$.
2. Claim 2. For each step t and each $u \in \mathit{frontier}_t(\textsc{ShortestPath}_1)$ there exists a $t' \geq t$ such that $u \in \mathit{frontier}_{t'}(\textsc{ShortestPath}_2)$.

Here we denote by exp_t the value of exp in the state reached by t steps. An index $i \in \{1, 2\}$ in $exp_t(i)$ refers to the value in a state of $\textsc{ShortestPath}_i$.

Corollary. $\textsc{ShortestPath}_i$ for $i = 1, 2$ label the same nodes as $visited$, once. Thus, the refinement preserves *Correctness and Termination*.

Slow Down Lemma: Proof by induction on runs

- $t = 0$: $\textsc{ShortestPath}_i$ both have $\mathit{frontier}_0(i) = \{\mathit{source}\}$
- $t \Rightarrow t + 1$:
  - ad claim 1:
    - Case 1. Let $v \in \mathit{frontier}_t(2)$. Then by induction hypothesis $v \in \mathit{frontier}_{t'}(1)$ for some $t' \leq t$.
    - Case 2. Let $v \in \mathit{frontier}_{t+1}(2) \setminus \mathit{frontier}_t(2)$.
      Let $u \in \mathit{frontier}_t(2)$ be the element chosen by step $t + 1$ of $\textsc{ShortestPath}_2$. By case 1, $u \in \mathit{frontier}_{t'}(1)$ for some $t' \leq t$.
      Then after the next step of $\textsc{ShortestPath}_1$, namely $t' + 1 \leq t + 1$, each element of $\mathit{neighb}(u)$ is $\mathit{visited}$, including $v \in \mathit{neighb}(u)$.
      Hence, either v has been labeled as $\mathit{visited}(1)$ already before step $t' + 1$, so that $v \in \mathit{frontier}_{t''}(1)$ for some $t'' \leq t'$, or v is 'last labeled' as $\mathit{visited}$ by step $t' + 1$ of $\textsc{ShortestPath}_1$, which implies $v \in \mathit{frontier}_{t'+1}(\textsc{ShortestPath}_1)$.

Slow Down Lemma: Proof by induction on runs (Cont'd)

- ad claim 2: By definition of $\textsc{ShortestPath}_1$, the following equation holds:

$$frontier_{t+1}(1) = \bigcup_{u \in frontier_t(1)} neighb(u) \setminus Visited_t(1).$$

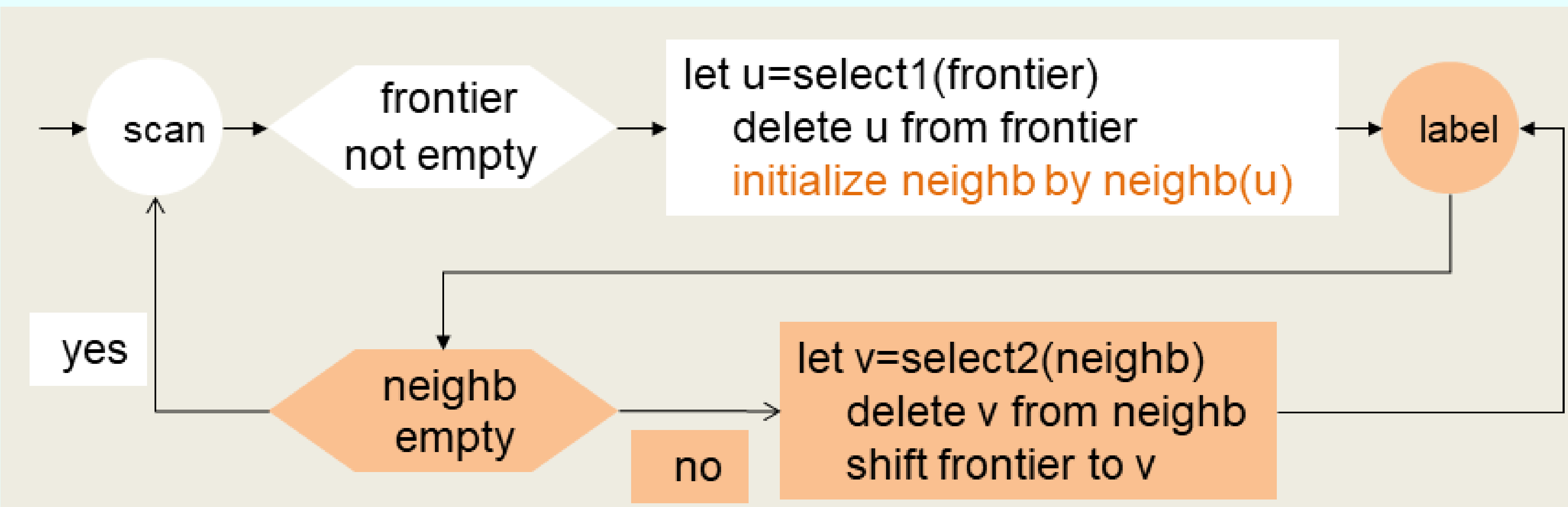
- For every $u \in frontier_t(1)$ the induction hypothesis implies $u \in frontier_{t'}(2)$ for some $t' \geq t$.
- Let $u \in frontier_{t'}(2)$ be chosen by $\textsc{ShortestPath}_2$ in step $t' + 1$. For each $v \in neighb(u) \setminus Visited_t(1)$ this step yields $v \in frontier_{t'+1}(2)$.

Third refinement: iterative implementation of 'forall'

Idea: edgewise frontier propagation, implementing **forall** in $\textsc{ShiftFrontierToNeighb}(u)$ by an iterating submachine

- initialize $neighb = neighb(u)$ and then select one by one nodes v of $neighb$ to edgewise $\textsc{ShiftFrontierTo}(v)$ until $neighb = \emptyset$

$\textsc{ShortestPath}_3 =$



Refinement correctness for edgewise frontier propagation

- each iteration segment of successive $\textsc{ShortestPath}_3$ steps which
  - first choose an $u \in \mathit{frontier}$
  - then choose successively all and only the neighbors v of u
    - to apply $\textsc{ShiftFrontierTo}(v)$
- corresponds and is equivalent to one $\textsc{ShortestPath}_2$ step
  - which applies to the same $u \in \mathit{frontier}$ in one $\textsc{ShiftFrontierToNeighb}(u)$ step simultaneously $\textsc{ShiftFrontierTo}(v)$ for all $v \in \mathit{neighb}(u)$

This is an example of a $(1, \mathit{many})$ refinement, with u-dependent values of many, defined by the graph structure:

$$\mathit{many} = 1 + \mathit{outFan}(u)$$

Corollary. The refinement of $\textsc{ShortestPath}_2$ to $\textsc{ShortestPath}_3$ preserves *Correctness and Termination*.

Fourth refinement: implementing data structures

$\textsc{ShortestPath}_4$ refines two data structures:

- $\mathit{frontier}$ is refined by a *queue*: $select1 = fst$, $\textsc{Delete}$ at one end and $\textsc{Append}$ at the other end
- neighb is refined to a *stack*: $select2 = top$, $\textsc{Delete} = pop$

This is an example of a pure data refinement:

- the steps in the runs are in 1-1-correspondence

Refinement Correctness of this 1-1 refinement boils down to the correctness of the well-known set implementations by queues resp. stacks.

Corollary. The data refinement of $\textsc{ShortestPath}_3$ to $\textsc{ShortestPath}_4$ preserves *Correctness and Termination*.

Introducing weights to measure paths frou $source$ to u

Extend a given edge $weight\colon E \to \mathbb{R}^+$ to a path $weight$ as follows:

$$weight(\epsilon) = 0, weight(pe) = weight(p) + weight(e)$$

$$minWeight(u) = inf\{weight(p) \mid p \text{ is a path from } source \text{ to } u\}$$

NB. Instead of $\mathbb{R}^+$, any well-founded partial order $(M, <)$ of path measures works which has the following properties:

- there are a smallest and a largest element 0 resp. ∞
- any $m, m' \in M$ have an infimum (greatest lower bound)
- adding edge weight to path measures is monotonic wrt path measures and distributive wrt inf, i.e. for each $m, m' \in M$ and edge weight w:

$-\, m < m'$ implies $m + w < m' + w$

$-\, inf(X) + w = inf\{x + w \mid x \in X\}$

Fifth refinement: compute minimal path from $source$ to u

Goal: when visiting u from $source$, compute also a minimal path, i.e. of minimal $minWeight(u)$, along the (possibly multiple) paths

- by *successive approximations of an upper bound* $upbd\colon NODE \to \mathbb{R}$
- starting with $upbd(u) = \infty$ for every node u, except $upbd(source) = 0$

Idea: Refine $\textsc{ShiftFrontierTo}$, along an edge e from u to v

- trying to lower $upbd(v)$ to $upbd(u) + weight(e)$

Problem: feature interaction of conflicting (not purely incremental) requirements, namely:

- each node is visited only once
- compute a minimal path from source to each reachable node by stepwise improving approximations, possibly discovering a shorter path upon revisiting the node

Conflict resolution: revisit nodes if it improves their $upbd$

Conflict resolution: Each time $upbd(v)$, for a path from $source$ to v, $CanBeLoweredBy$ a path going through an edge (u, v) from an already $visited$ neighbor $u \in frontier$, v is $\textsc{Insert}$ed into $frontier$:

- When $v \neq source$ is $visited$ for the first time, say via an already $visited$ neighbor node $u \in frontier$ (so that $upbd(u) < \infty$), its $upbd(v) = \infty$ $CanBeLoweredBy$ updating it using $upbd(u) + weight(u, v)$.
- When $upbd(v) < \infty$ (so that v has already been $visited$) but $upbd(v)$ $CanBeLoweredBy$ a path going through a neighbor node $u \in frontier$, v is 'revisited'
  – meaning that it is $\textsc{Insert}$ed once more into $frontier$.

Refinement implementing conflict resolution

$\textsc{ShortestPath}_5$ is $\textsc{ShortestPath}_4$ refined as follows:

- Add $currSource := u$ to $neighb := neighb(u)$ initialization
- $\textsc{ShiftFrontierTo}(v) =$

  if $v \notin visited$ **then** -- upon first visit $upbd(v) = \infty$ holds

  $visited(v) := true$

  $\textsc{Insert}(v, frontier)$

  $\textsc{LowerUpbd}(upbd(v), (currSource, v))$ -- yields $upbd(v) < \infty$

  -- because $u \in frontier$ implies $upbd(u) < \infty$

  else

  if $CanBeLoweredBy(upbd(v), (currSource, v))$ **then**

  $\textsc{LowerUpbd}(upbd(v), (currSource, v))$

  $\textsc{Insert}(v, frontier)$ -- neighbors may have to $\textsc{LowerUpbd}$

Meaning of $\textsc{LowerUpbd}$

$CanBeLoweredBy(bd, (u, v))$ **iff**

$upbd(u) + weight(u, v) < bd$ -- Dijkstra's algorithm where $M = \mathbb{R}$

$bd \not\leq upbd(u) + weight(u, v)$ -- Moore's algorithm

$\textsc{LowerUpbd}(bd, (u, v)) =$

$bd := upbd(u) + weight(u, v)$ -- Dijkstra

$bd := inf\{bd, upbd(u) + weight(u, v)\}$ -- Moore

Remark. A further refinement step could restrict $frontier$ to a *priority queue*, selecting nodes with least $upbd$:

$u = select1(frontier)$ **iff forall** $v \in frontier \;\; upbd(u) \leq upbd(v)$

Refinement is labeling correct and complete

- **Completeness**: by definition, $\textsc{ShortestPath}_5$ is a purely incremental extension, also called *conservative refinement*, of $\textsc{ShortestPath}_4$.
  - In fact, each $\textsc{ShortestPath}_4$ step corresponds to a step of $\textsc{ShortestPath}_5$ with equivalent labeling:
    - $select1$-steps choosing an u for the first time in $frontier$
    - $select2$-steps with a first-visit application of $\textsc{ShiftFrontierTo}$
- **Correctness**: for every pair (r_4, r_5) of corresponding $\textsc{ShortestPath}_i$ runs ($i = 4, 5$),
  - i.e. runs started in the same initial state and with (where corresponding) same $select$ions

projecting from r_5 a) $select1$-steps which choose an u that is for the first time in $frontier$, and b) $select2$-steps with a first-visit application of $\textsc{ShiftFrontierTo}$ (not considering the $\textsc{LowerUpdb}$ submachine) yields r_4.

Termination of $\textsc{ShortestPath}_5$

Since elements may be reinserted into $frontier$, to prove the termination property it has to be shown that eventually $frontier$ becomes empty.

Each time in mode $scan$ an element $u \in frontier$ is selected, at the end of the iteration, when $mode$ becomes again $scan$, the following holds:

- either $frontier$ is decreased by 1
  - namely if for none of u's neighbors v $CanBeLoweredBy(upbd(v), (u, v))$ so that the sum of $upbd(v)$ of u's neighbors v remains unchanged
- or the number of u's neighbors v with $upbd(v) = \infty$ or the sum of $upbd(v) < \infty$ of u's neighbors v is decreased
  - which can happen only finitely often

Correctness of Shortest-Path-Property for Moore's algorithm

Theorem. When $\textsc{ShortestPath}_5$ terminates, for every u holds:

$$minWeight(u) = upbd(u)$$

The proof follows from two lemmata:

- **Lemma 1**. $minWeight(u) \leq upbd(u)_t$ holds for each u after each step t.
- **Lemma 2**. When $\textsc{ShortestPath}_5$ terminates, $upbd(u) \leq weight(p)$ holds for every path p from $source$ to u.

In fact, let t be the last step of $\textsc{ShortestPath}_5$. Then

$minWeight(u) \leq upbd(u)_t$ -- by Lemma 1

$\leq weight(p)$ -- by Lemma 2 for each $source$-to-u path p

Thus, $upbd(u)_t$ is a lower bound of $weight(p)$ for every $source$-to-u path p, therefore $upbd(u)_t \leq minWeight(u)$, the greatest such bound. Hence $minWeight(u) = upbd(u)_t$.

Proof of Lemma 1 ($minWeight(u) \leq upbd(u)_t$) by induction

- $t = 0$: claim holds by definition of $upbd(source) = 0$ and $upbd(u) = \infty$ for each $u \neq source$
- if in step $t+1$ $upbd(v)$ is updated, then to $inf\{upbd(v)_t, upbd(u)_t + weight(u,v)\}$. Since $minWeight(v)$ is a lower bound for both values in the set (see below), it is $\leq$ their greatest lower bound.
  - $minWeight(v) \leq upbd(v)_t$ holds by ind.hyp.
  - $minWeight(v) \leq minWeight(u) + weight(u,v)$ (see below) and ind.hyp. $minWeight(u) \leq upbd(u)_t$ imply the claim.
    - $minWeight(v)$
      $=_{Def}$ $inf\{weight(p) \mid p$ path from $source$ to $v\}$
      $\leq$ $inf\{weight(p.(u,v)) \mid p$ path from $source$ to $u\}$
      $=_{Def}$ $\inf\{weight(p) + weight(u,v) \mid p$ path from $source$ to $u\}$
      $=_{Distr}$ $\inf\{weight(p) \mid p$ path from $source$ to $u\} + weight(u,v)$
      $=_{Def}$ $minWeight(u) + weight(u,v)$

Proof of Lemma 2 ($upbd(u) \le weight(p)$ upon termination)

Induction on path length t:

- $t = 0$: claim follows from $upbd(source) = 0 = weight(\epsilon)$
- Let $p.(u, v)$ be any path of length $t + 1$.
  - $upbd(v) \le upbd(u) + weight(u, v)$
    - otherwise $\textsc{LowerUpbd}(upbd(v), u)$ could fire
  - $upbd(u) \le weight(p)$ by ind.hyp.

  Therefore

  $upbd(v) \le upbd(u) + weight(u, v)$

  $\le weight(p) + weight(u, v)$ // by monotonicity

  $= weight(p.(u, v))$ // by definition of path $weight$

Refinement by a rule optimization

NB. In $\textsc{ShortestPath}_5$ the following equivalence holds:

$$visited(u) \textbf{ iff } upbd(u) < \infty$$

Therefore the IF-clause updates in $\textsc{ShiftFrontierTo}(v)$:

if $v \notin visited$ **then** -- upon first visit $upbd(v) = \infty$ holds

$visited(v) := true$ $\quad$ $\textsc{Insert}(v, frontier)$

$\textsc{LowerUpbd}(upbd(v), (currSource, v))$ -- yields $upbd(v) < \infty$

-- because $u \in frontier$ implies $upbd(u) < \infty$

are also performed by the updates in the ELSE-clause:

if $CanBeLoweredBy(upbd(v), (currSource, v))$ **then**

$\textsc{LowerUpbd}(upbd(v), (currSource, v))$

$\textsc{Insert}(v, frontier)$ -- neighbors may have to $\textsc{LowerUpbd}$

so that the rule can be optimized to this ELSE-clause.[2]

[2] Suggestion made by Mario Wenzel (Halle), Oct'19.

Refinement to C^{++} code

- It remains to instantiate data structures for measures and weights
- See: K. Stroetmann: The Constrained Shortest Path Problem: A Case Study in Using ASMs.
  - J. of Universal Computer Science 3 (4), 1997.

References

- *E. F. Moore: The Shortest Path Through a Maze.* Proc. Intern. Symp. on the Theory of Switching, Part II, Vol. 30 of "The Annals of the Computation Laboratory of Harvard University", Cambridge, MA, 1959, Harvard University Press.
- *K. Stroetmann: The Constrained Shortest Path Problem: A Case Study in Using ASMs*. In: J. of Universal Computer Science 3 (4), 1997.
- E. Börger and R. Stärk: Abstract State Machines. Springer 2003. See Ch.3.2.1 for a correctness proof for Dijkstra's algorithm $\textsc{ShortestPath}_5$.
- E. Börger and A. Raschke: Modeling Companion for Software Practitioners. Springer 2018 http://modelingbook.informatik.uni-ulm.de

Copyright Notice